

9 OPL AND DISKS

Using disks in OPL

Î On the Series 5, *memory disks* in the D: drive may be used in the same way as the internal memory by specifying D: where you would usually specify C:.

Î On the Series 3c, Solid State Disks (SSDs), which are explained in detail in the User guide, may be used.

Siena On the Siena, there is no support for using disks.

There are two main reasons for using disks:

- To provide more room for storing data.
- To make backup copies of important information, in case you accidentally change or delete it (or even lose your Psion).

The discussion below explains the use of SSDs for OPL on the Series 3c.

Î Types of Solid State Disk

There are two types - *Ram SSDs* and *Flash SSDs*. They fit into the SSD drives marked A and B, at either side of the Series 3c.

- Flash SSDs are for storing or backing up information which is infrequently changed. This includes finished OPL programs.
- Ram SSDs are for storing or backing up information which changes frequently. This includes OPL programs you are still writing or testing.

You can, though, save programs and data files to either kind of SSD, as you see fit.

Î How to put programs on an SSD

To create a new OPL module on an SSD, use the 'New file' option in the System screen as before, but set the Disk line of the dialog to A or B as required.

To copy an OPL module on to an SSD, move onto the module name where it is listed under the Program icon, and use the 'Copy file' option on the 'File' menu. Set the 'To file: Disk' line to A or B. If you want this copy to have a different name to the original, type the name to use, on the 'To file: Name' line. The new copy will appear in the list under the Program icon, but with [A] or [B] after its name.

To copy the **translated** version of an OPL module, move onto the name in the list under the RunOpl icon (to the right of the Program icon), then proceed as before.

Î SSDs from inside OPL

Your OPL programs can create or use data files on SSDs. To do so, begin the name of the data file with A: or B: - for example:

```
CREATE "B:JKQ",A,X1$,X2$
```

tries to create a data file JKQ on an SSD in B, while

```
DELETE "A:X300"
```

tries to delete a data file X300 on an SSD in A.

Don't confuse the drive names A and B with the logical names A, B, C and D. Logical names are unaffected by which drive a data file is on.

The internal memory can be referred to as M:, if required. For example:

```
PROC delx300:
```

```

LOCAL a$(3),c%
a$="MAB" :c%=1          REM default to "Internal"
dINIT "Delete X300 data file"
dCHOICE c%,"Disk:", "Internal,A,B"
IF DIALOG                REM returns 0 if cancelled
    DELETE MID$(A$,c%,1)+":X300"
ENDIF
ENDP

```

In this example, `MID$(A$,c%,1)` gives "M", "A" or "B", according to the choice made in the dialog. This is added on the front of `:X300` to give the name of the file to delete - "M:X300", "A:X300" or "B:X300".

When using data files with SSDs, follow the same guidelines as with OPL programs - Flash SSDs are for one-off or "finished" information, while Ram SSDs are for information which is still being changed.

Directories and DOS structure

The internal memory, memory disks and SSDs use a DOS-compatible directory structure, the same as that used by disks on PCs. For more details, see the 'Advanced Topics' chapter.